# Python Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

## Creating a Function

In Python a function is defined using the def keyword:

### Example

def my\_function():

print("Hello from a function")

## Calling a Function

To call a function, use the function name followed by parenthesis:

### Example

def my\_function():

print("Hello from a function")

**my\_function()**

## Parameters

Information can be passed to functions as parameter.

Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a function with one parameter (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

### Example

def my\_function(**fname**):

print(fname + " Refsnes")

my\_function(**"Emil"**)

my\_function(**"Tobias"**)

my\_function(**"Linus"**)

## Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without parameter, it uses the default value:

### Example

def my\_function(**country = "Norway"**):

print("I am from " + country)

my\_function("Sweden")

my\_function("India")

my\_function()

my\_function("Brazil")

## Passing a List as a Parameter

You can send any data types of parameter to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as a parameter, it will still be a List when it reaches the function:

### Example

def my\_function(food):

for x in food:

print(x)

fruits = ["apple", "banana", "cherry"]

my\_function(fruits)

## Return Values

To let a function return a value, use the return statement:

### Example

def my\_function(x):

**return 5 \* x**

print(my\_function(3))

print(my\_function(5))

print(my\_function(9))

## Recursion

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

In this example, tri\_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out how exactly this works, best way to find out is by testing and modifying it.

### Example

Recursion Example

def tri\_recursion(k):

if(k>0):

result = k+tri\_recursion(k-1)

print(result)

else:

result = 0

return result

print("\n\nRecursion Example Results")

tri\_recursion(6)

## Test Yourself With Exercises

## Exercise:

Create a function named my\_function.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ :

print("Hello from a function")